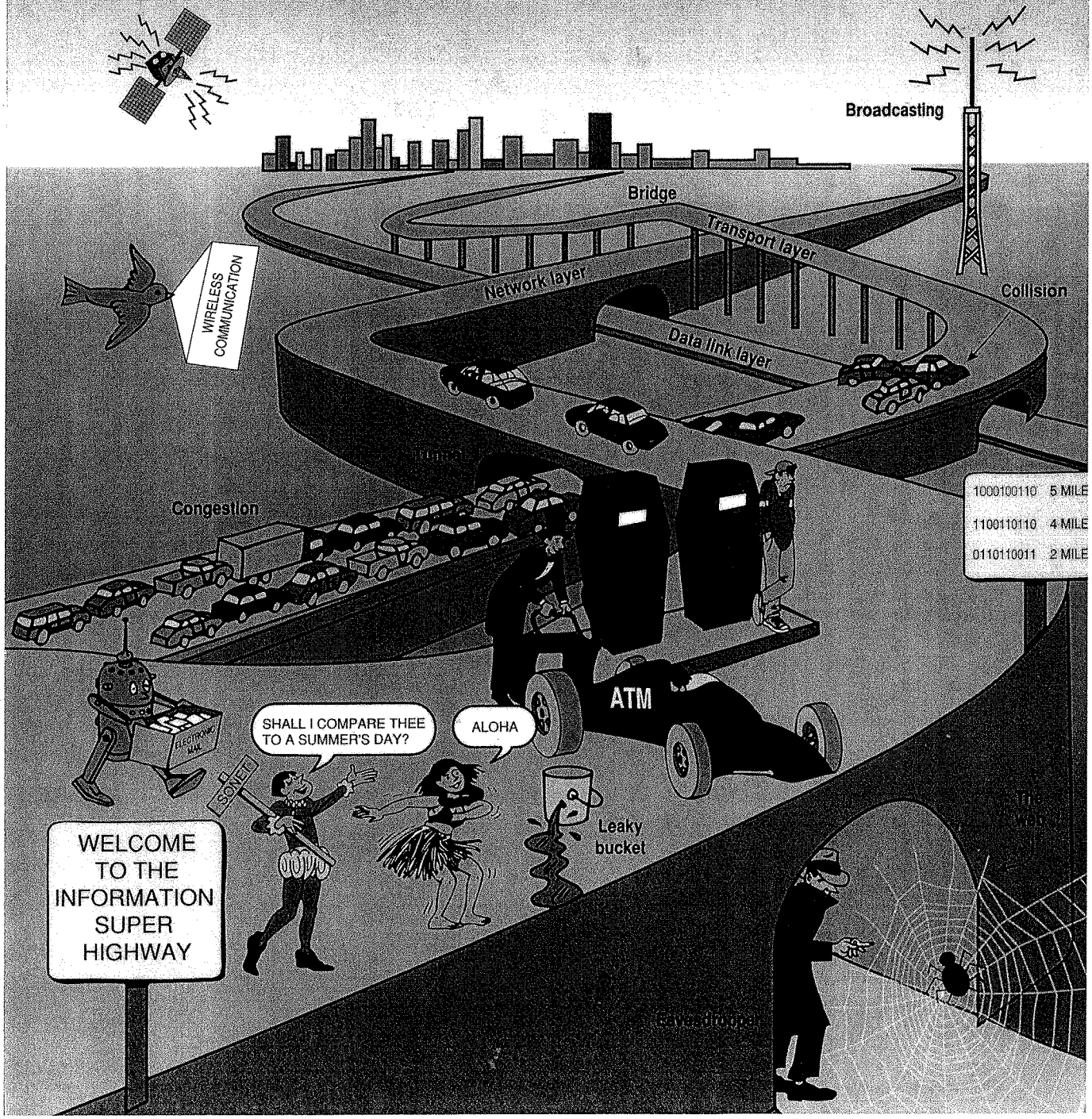
EXHIBIT 5

THIRD EDITION

COMPUTER NETWORKS

ANDREW S. TANENBAUM



Library of Congress Cataloging in Publication Data

Tanenbaum, Andrew S. 1944-.
Computer networks / Andrew S. Tanenbaum. -- 3rd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-349945-6
1.Computer networks. I. Title.
TK5105.5.T36 1996 96-4121
004.6--dc20 CIP

Editorial/production manager: *Camille Trentacoste*
Interior design and composition: *Andrew S. Tanenbaum*
Cover design director: *Jerry Votta*
Cover designer: *Don Martinetti, DM Graphics, Inc.*
Cover concept: *Andrew S. Tanenbaum, from an idea by Marilyn Tremaine*
Interior graphics: *Hadel Studio*
Manufacturing manager: *Alexis R. Heydt*
Acquisitions editor: *Mary Franz*
Editorial Assistant: *Noreen Regina*

Prentice-Hall, Inc.
A Simon & Schuster Company
Upper Saddle River, New Jersey 07458

The publisher offers discounts on this book when ordered in bulk quantities. For more information, contact:
Corporate Sales Department, Prentice Hall PTR, One Lake Street, Upper Saddle River, NJ 07458.
Phone: (800) 382-3419; Fax: (201) 236-7141. E-mail: corpsales@prenhall.com

Printed in the United States of America
10 9

ISBN 0-13-349945-6

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

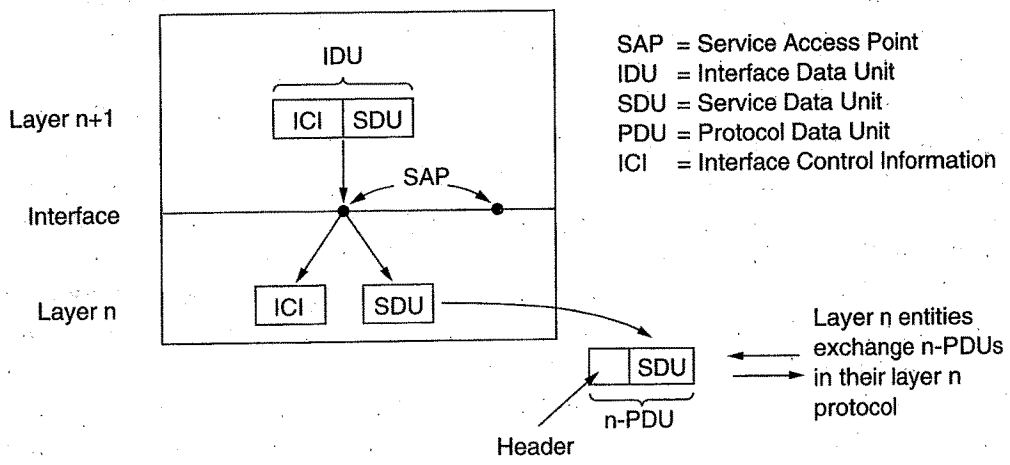


Fig. 1-12. Relation between layers at an interface.

to carry out their peer protocol. They identify which PDUs contain data and which contain control information, provide sequence numbers and counts, and so on.

1.3.4. Connection-Oriented and Connectionless Services

Layers can offer two different types of service to the layers above them: connection-oriented and connectionless. In this section we will look at these two types and examine the differences between them.

Connection-oriented service is modeled after the telephone system. To talk to someone, you pick up the phone, dial the number, talk, and then hang up. Similarly, to use a connection-oriented network service, the service user first establishes a connection, uses the connection, and then releases the connection. The essential aspect of a connection is that it acts like a tube: the sender pushes objects (bits) in at one end, and the receiver takes them out in the same order at the other end.

In contrast, **connectionless service** is modeled after the postal system. Each message (letter) carries the full destination address, and each one is routed through the system independent of all the others. Normally, when two messages are sent to the same destination, the first one sent will be the first one to arrive. However, it is possible that the first one sent can be delayed so that the second one arrives first. With a connection-oriented service this is impossible.

Each service can be characterized by a **quality of service**. Some services are reliable in the sense that they never lose data. Usually, a reliable service is implemented by having the receiver acknowledge the receipt of each message, so the sender is sure that it arrived. The acknowledgement process introduces overhead and delays, which are often worth it but are sometimes undesirable.

A typical situation in which a reliable connection-oriented service is

appropriate is file transfer. The owner of the file wants to be sure that all the bits arrive correctly and in the same order they were sent. Very few file transfer customers would prefer a service that occasionally scrambles or loses a few bits, even if it is much faster.

Reliable connection-oriented service has two minor variations: message sequences and byte streams. In the former, the message boundaries are preserved. When two 1-KB messages are sent, they arrive as two distinct 1-KB messages, never as one 2-KB message. (Note: KB means kilobytes; kb means kilobits.) In the latter, the connection is simply a stream of bytes, with no message boundaries. When 2K bytes arrive at the receiver, there is no way to tell if they were sent as one 2-KB message, two 1-KB messages, or 2048 1-byte messages. If the pages of a book are sent over a network to a phototypesetter as separate messages, it might be important to preserve the message boundaries. On the other hand, with a terminal logging into a remote timesharing system, a byte stream from the terminal to the computer is all that is needed.

As mentioned above, for some applications, the delays introduced by acknowledgements are unacceptable. One such application is digitized voice traffic. It is preferable for telephone users to hear a bit of noise on the line or a garbled word from time to time than to introduce a delay to wait for acknowledgements. Similarly, when transmitting a video film, having a few pixels wrong is no problem, but having the film jerk along as the flow stops to correct errors is very irritating.

Not all applications require connections. For example, as electronic mail becomes more common, can electronic junk mail be far behind? The electronic junk mail sender probably does not want to go to the trouble of setting up and later tearing down a connection just to send one item. Nor is 100 percent reliable delivery essential, especially if it costs more. All that is needed is a way to send a single message that has a high probability of arrival, but no guarantee. Unreliable (meaning not acknowledged) connectionless service is often called **datagram service**, in analogy with telegram service, which also does not provide an acknowledgement back to the sender.

In other situations, the convenience of not having to establish a connection to send one short message is desired, but reliability is essential. The **acknowledged datagram service** can be provided for these applications. It is like sending a registered letter and requesting a return receipt. When the receipt comes back, the sender is absolutely sure that the letter was delivered to the intended party and not lost along the way.

Still another service is the **request-reply service**. In this service the sender transmits a single datagram containing a request; the reply contains the answer. For example, a query to the local library asking where Uighur is spoken falls into this category. Request-reply is commonly used to implement communication in the client-server model: the client issues a request and the server responds to it. Figure 1-13 summarizes the types of services discussed above.

	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Remote login
	Unreliable connection	Digitized voice
Connectionless	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Registered mail
	Request-reply	Database query

Fig. 1-13. Six different types of service.

1.3.5. Service Primitives

A service is formally specified by a set of **primitives** (operations) available to a user or other entity to access the service. These primitives tell the service to perform some action or report on an action taken by a peer entity. One way to classify the service primitives is to divide them into four classes as shown in Fig. 1-14.

Primitive	Meaning
Request	An entity wants the service to do some work
Indication	An entity is to be informed about an event
Response	An entity wants to respond to an event
Confirm	The response to an earlier request has come back

Fig. 1-14. Four classes of service primitives.

To illustrate the uses of the primitives, consider how a connection is established and released. The initiating entity does a CONNECT.request which results in a packet being sent. The receiver then gets a CONNECT.indication announcing that an entity somewhere wants to set up a connection to it. The entity getting the CONNECT.indication then uses the CONNECT.response primitive to tell whether it wants to accept or reject the proposed connection. Either way, the entity issuing the initial CONNECT.request finds out what happened via a CONNECT.confirm primitive.

Primitives can have parameters, and most of them do. The parameters to a CONNECT.request might specify the machine to connect to, the type of service desired, and the maximum message size to be used on the connection. The parameters to a CONNECT.indication might contain the caller's identity, the type of

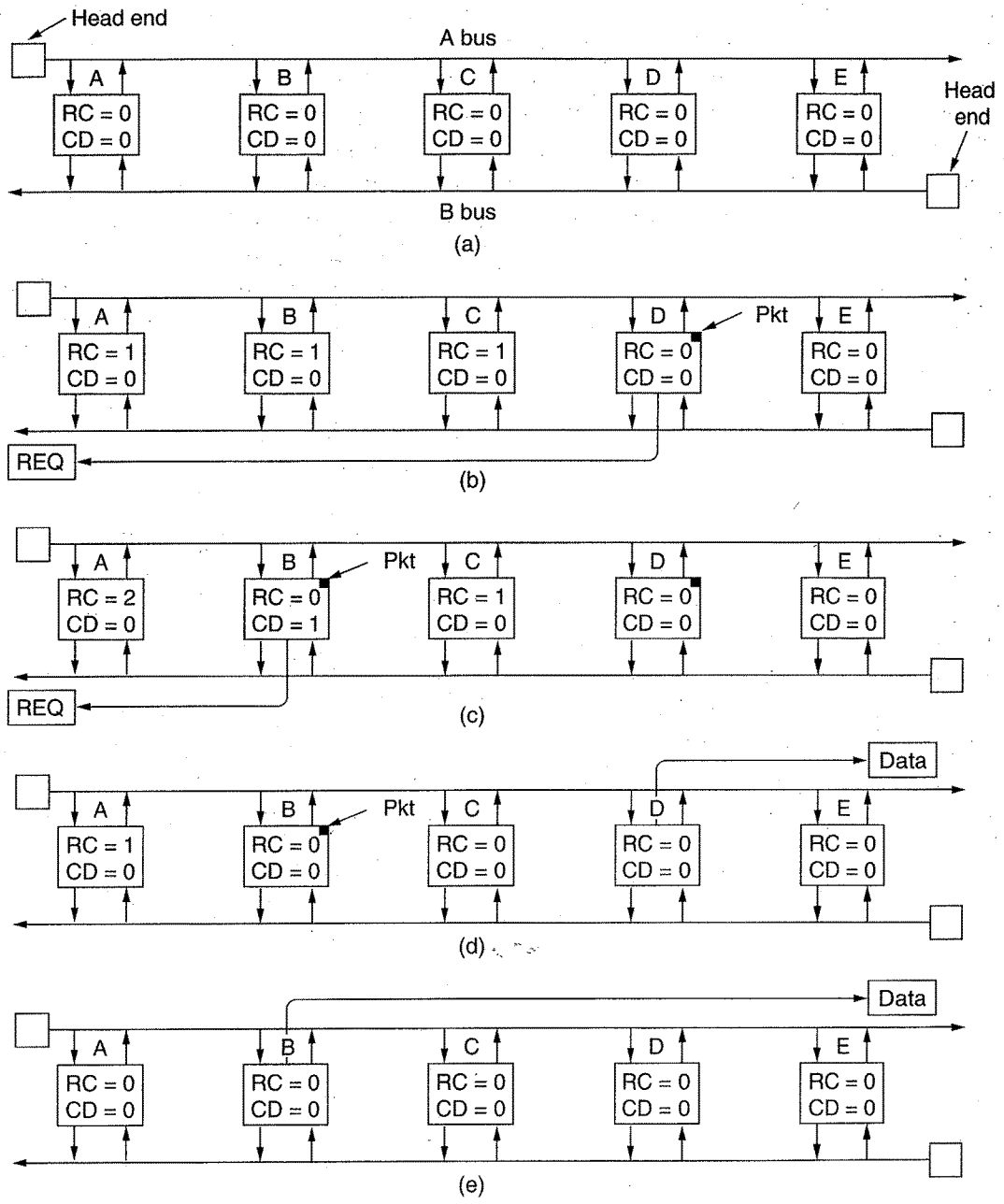


Fig. 4-32. (a) Initially the MAN is idle. (b) After *D* makes a request. (c) After *B* makes a request. (d) After *D* transmits. (e) After *B* transmits.

data link protocols. These protocols provided error control (using acknowledgements) and flow control (using a sliding window).

In contrast, in this chapter, we have not said a word about reliable communication. All that the 802 LANs and MAN offer is a best-efforts datagram service.

Sometimes, this service is adequate. For example, for transporting IP packets, no guarantees are required or even expected. An IP packet can just be inserted into an 802 payload field and sent on its way. If it gets lost, so be it.

Nevertheless, there are also systems in which an error-controlled, flow-controlled data link protocol is desired. IEEE has defined one that can run on top of all the 802 LAN and MAN protocols. In addition, this protocol, called **LLC** (**Logical Link Control**), hides the differences between the various kinds of 802 networks by providing a single format and interface to the network layer. This format, interface, and protocol are all closely based on OSI. LLC forms the upper half of the data link layer, with the MAC sublayer below it, as shown in Fig. 4-33.

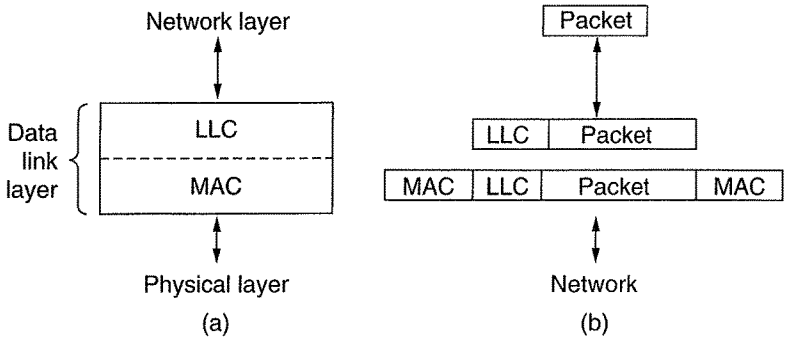


Fig. 4-33. (a) Position of LLC. (b) Protocol formats.

Typical usage of LLC is as follows. The network layer on the sending machine passes a packet to LLC using the LLC access primitives. The LLC sublayer then adds an LLC header, containing sequence and acknowledgement numbers. The resulting structure is then inserted into the payload field of an 802.x frame and transmitted. At the receiver, the reverse process takes place.

LLC provides three service options: unreliable datagram service, acknowledged datagram service, and reliable connection-oriented service. The LLC header is based on the older HDLC protocol. A variety of different formats are used for data and control. For acknowledged datagram or connection-oriented service, the data frames contain a source address, a destination address, a sequence number, an acknowledgement number, and a few miscellaneous bits. For unreliable datagram service, the sequence number and acknowledgement number are omitted.

4.4. BRIDGES

Many organizations have multiple LANs and wish to connect them. LANs can be connected by devices called **bridges**, which operate in the data link layer. This statement means that bridges do not examine the network layer header and

can thus copy IP, IPX, and OSI packets equally well. In contrast, a pure IP, IPX, or OSI router can handle only its own native packets.

In the following sections we will look at bridge design, especially for connecting 802.3, 802.4, and 802.5 LANs. For a comprehensive treatment of bridges and related topics, see (Perlman, 1992). Before getting into the technology of bridges, it is worthwhile taking a look at some common situations in which bridges are used. We will mention six reasons why a single organization may end up with multiple LANs. First, many university and corporate departments have their own LANs, primarily to connect their own personal computers, workstations, and servers. Since the goals of the various departments differ, different departments choose different LANs, without regard to what other departments are doing. Sooner or later, there is a need for interaction, so bridges are needed. In this example, multiple LANs came into existence due to the autonomy of their owners.

Second, the organization may be geographically spread over several buildings separated by considerable distances. It may be cheaper to have separate LANs in each building and connect them with bridges and infrared links than to run a single coaxial cable over the entire site.

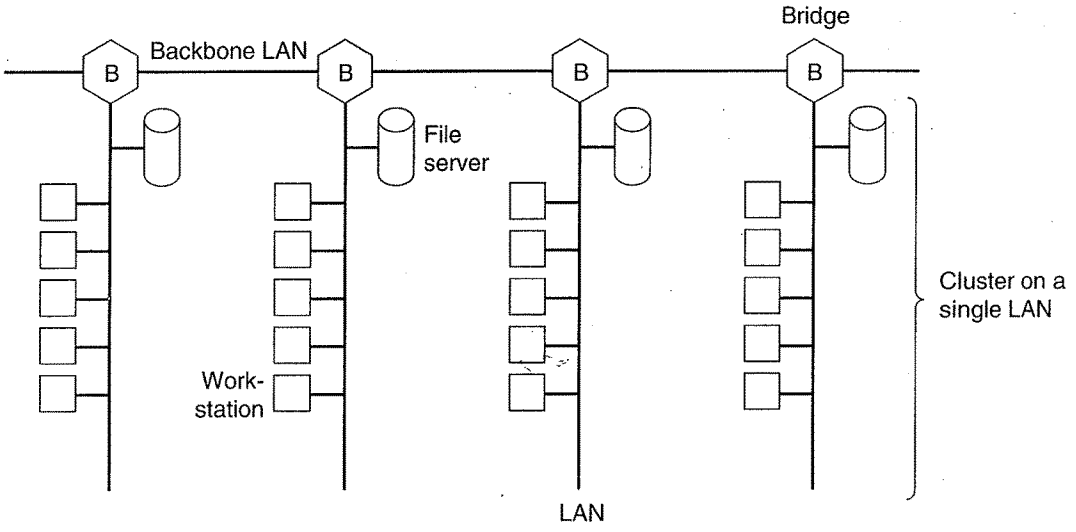


Fig. 4-34. Multiple LANs connected by a backbone to handle a total load higher than the capacity of a single LAN.

Third, it may be necessary to split what is logically a single LAN into separate LANs to accommodate the load. At many universities, for example, thousands of workstations are available for student and faculty computing. Files are normally kept on file server machines, and are downloaded to users' machines upon request. The enormous scale of this system precludes putting all the workstations on a single LAN—the total bandwidth needed is far too high. Instead multiple LANs connected by bridges are used, as shown in Fig. 4-34. Each LAN

contains a cluster of workstations with its own file server, so that most traffic is restricted to a single LAN and does not add load to the backbone.

Fourth, in some situations, a single LAN would be adequate in terms of the load, but the physical distance between the most distant machines is too great (e.g., more than 2.5 km for 802.3). Even if laying the cable is easy to do, the network would not work due to the excessively long round-trip delay. The only solution is to partition the LAN and install bridges between the segments. Using bridges, the total physical distance covered can be increased.

Fifth, there is the matter of reliability. On a single LAN, a defective node that keeps outputting a continuous stream of garbage will cripple the LAN. Bridges can be inserted at critical places, like fire doors in a building, to prevent a single node which has gone berserk from bringing down the entire system. Unlike a repeater, which just copies whatever it sees, a bridge can be programmed to exercise some discretion about what it forwards and what it does not forward.

Sixth, and last, bridges can contribute to the organization's security. Most LAN interfaces have a **promiscuous mode**, in which *all* frames are given to the computer, not just those addressed to it. Spies and busybodies love this feature. By inserting bridges at various places and being careful not to forward sensitive traffic, it is possible to isolate parts of the network so that its traffic cannot escape and fall into the wrong hands.

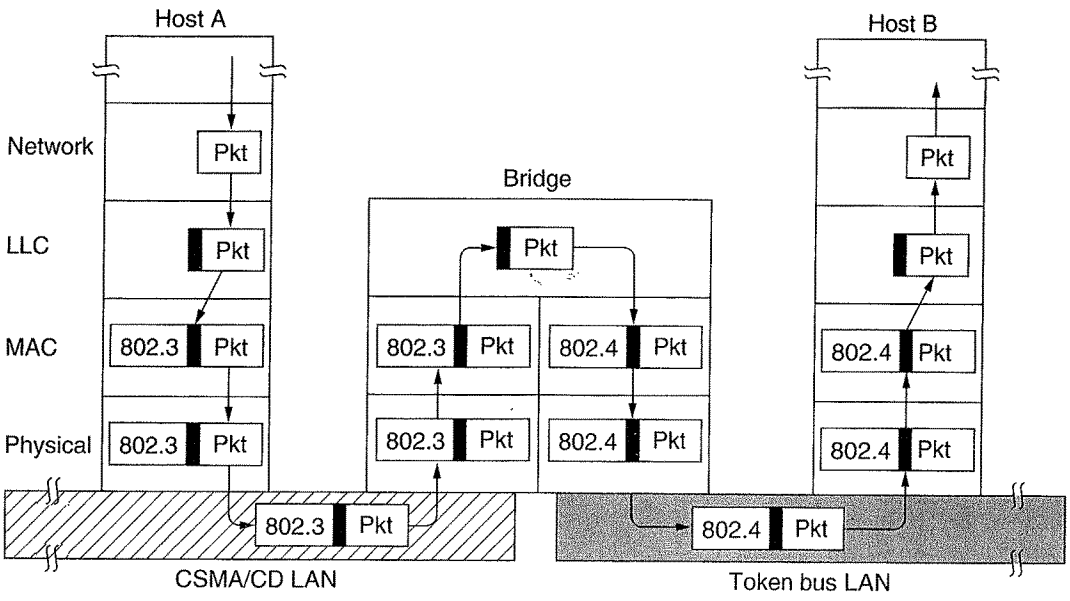


Fig. 4-35. Operation of a LAN bridge from 802.3 to 802.4.

Having seen why bridges are needed, let us now turn to the question of how they work. Figure 4-35 illustrates the operation of a simple two-port bridge. Host *A* has a packet to send. The packet descends into the LLC sublayer and acquires

an LLC header. Then it passes into the MAC sublayer and an 802.3 header is prepended to it (also a trailer, not shown in the figure). This unit goes out onto the cable and eventually is passed up to the MAC sublayer in the bridge, where the 802.3 header is stripped off. The bare packet (with LLC header) is then handed off to the LLC sublayer in the bridge. In this example, the packet is destined for an 802.4 subnet connected to the bridge, so it works its way down the 802.4 side of the bridge and off it goes. Note that a bridge connecting k different LANs will have k different MAC sublayers and k different physical layers, one for each type.

4.4.1. Bridges from 802.x to 802.y

You might naively think that a bridge from one 802 LAN to another one would be completely trivial. Such is not the case. In the remainder of this section we will point out some of the difficulties that will be encountered when trying to build a bridge between the various 802 LANs.

Each of the nine combinations of 802.x to 802.y has its own unique set of problems. However, before dealing with these one at a time, let us look at some general problems common to all the bridges. To start with, each of the LANs uses a different frame format (see Fig. 4-36). There is no valid technical reason for this incompatibility. It is just that none of the corporations supporting the three standards (Xerox, GM, and IBM) wanted to change *theirs*. As a result, any copying between different LANs requires reformatting, which takes CPU time, requires a new checksum calculation, and introduces the possibility of undetected errors due to bad bits in the bridge's memory. None of this would have been necessary if the three committees had been able to agree on a single format.

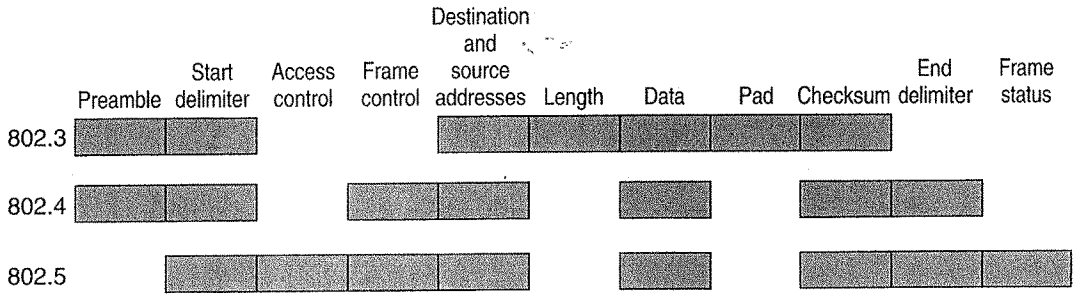


Fig. 4-36. The IEEE 802 frame formats.

A second problem is that interconnected LANs do not necessarily run at the same data rate. When forwarding a long run of back-to-back frames from a fast LAN to a slower one, the bridge will not be able to get rid of the frames as fast as they come in. It will have to buffer them, hoping not to run out of memory. The problem also exists from 802.4 to 802.3 at 10 Mbps to some extent because some